DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

//GAME DEVELOPMENT BASICS PART II

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

Gamedev Basics

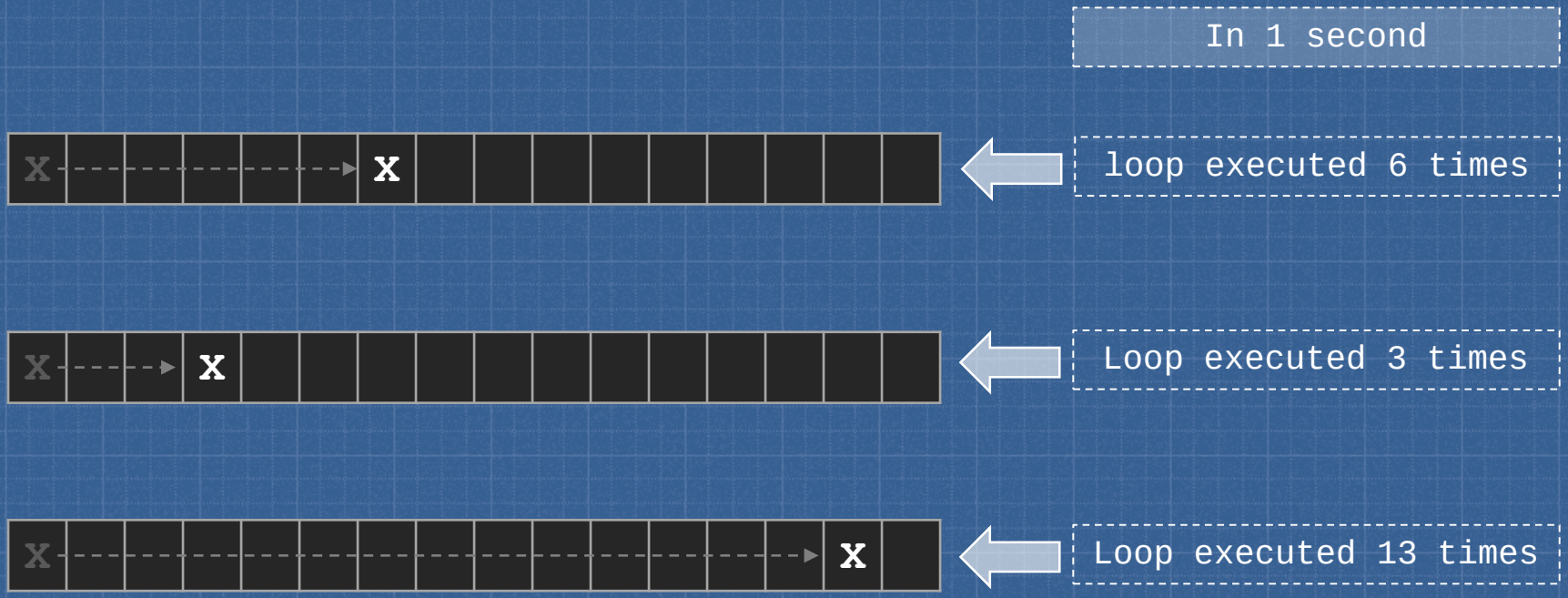
1. Part I
    1. Intro
    2. Frame
    3. Game Loop
    4. Ascii rendering
2. Part II
    1. **Clock**
    2. State Machine
3. Part III
    1. 1D array
    2. Collision Grid
    3. Code good Practices

The problem

```
#include "Console/Console.h"
#include <Windows.h>

static int bGameIsRunning = 1;
static int x = 0;

int main()
{
    // Game Initialization
    [Console initialization here]

    // Game Loop
    while (bGameIsRunning)
    {
        // Process Inputs
        [Input processing here]

        // Update
        x = x + 1;

        // Render
        Console_Clear();
        Console_Putchar(x,0,'X');
    }
    // Game Shutdown
    [Clean shutdown here]
}
```

- Current Code :
  - modify X
  - +1, each Frame

Game Loop time calculation is variable



Depending on Computer Speed, or Execution path, your Game loop can run faster or slower

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

How to fix it: Method 1

- Make sure the Game loop is always running at same speed
  - Insert artificial delay to keep it at X fps
  - “Frame locked”

FrameLocking

```
[includes here]
static int bGameIsRunning = 1;

int main()
{
    [Game initialization here]

    // Game Loop
    while (bGameIsRunning)
    {
        // Read Current Time at Frame Start
        double startTime = [GetCurrentTimeSeconds];

        [Input processing here]

        [Update here]

        [Render here]

        // Calculate the Time the Frame Took
        double frameTime = [GetCurrentTimeSeconds] - startTime;

        // Wait until 0.016666 sec elapsed for 60fps
        // Sleep is in Ms
        Sleep((0.016666 - frameTime)*1000);
    }

    [Game Shutdown here]
}
```

- We get the Start Time of a frame with a function
- We calculate the Duration of a Frame
- Sleep() the remaining of the 0.016666666…s to have a constant frame time

  (Or we do a Loop)

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

How to fix it: Method 2

- Modify the movement calculation to use time
  - DeltaTime: Time Elapsed between 2 frames
  - Euler Method movement Calculation

Euler method

Position =

Old Position+Velocity*DeltaTime

$$X = X_{-1} + V * dT;$$

Euler Method for Movement

```
#include "Console/Console.h“
#include "Clock/Clock.h“
#include <math.h> // C Math library
#include <Windows.h>

static int bGameIsRunning = 1;
static double eulerX = 0.f;
static double velocity = 0.001f; // Units per ms

int main()
{
    // Game Initialization
    [Console initialization here]

    // Game Loop
    while (bGameIsRunning)
    {
        // Frame Time calculation, DeltaTime
        Clock_GameLoopStart();

        // Process Inputs
        [Input processing here]

        // Update
        eulerX = eulerX + velocity * Clock_GetDeltaTime();

        // Render
        Console_Clear();
        Console_Putchar((int)round(eulerX),0,’X‘);
    }
    // Game Shutdown
    [Clean shutdown here]
}
```

- DeltaTime: milliseconds
  - Use Clock.h
- Need to Define a Velocity
  - units per ms
- Need to round the decimal value to integer value

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

Gamedev Basics

1. Part I
    1. Intro
    2. Frame
    3. Game Loop
    4. Ascii rendering
2. Part II
    1. Clock
    2. **State Machine**
3. Part III
    1. 1D array
    2. Collision Grid
    3. Code good Practices

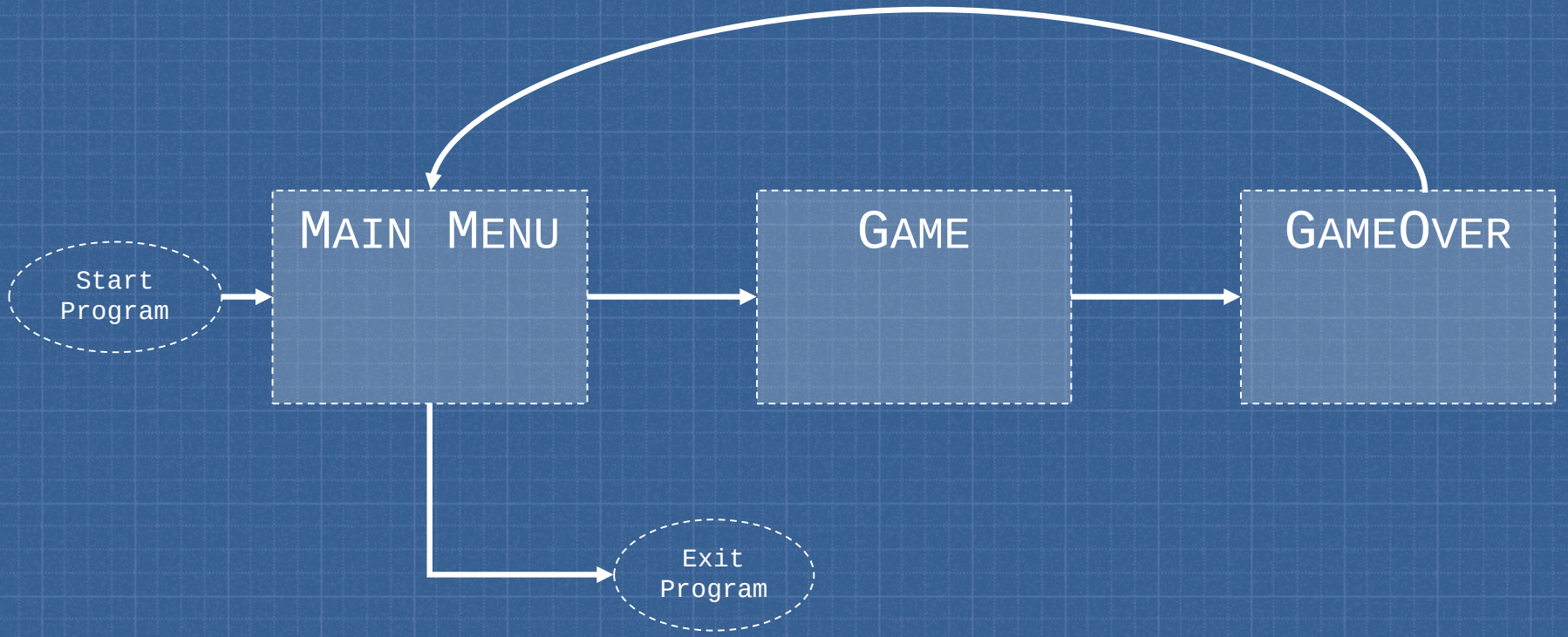
The problem

```
#include "Console/Console.h"
#include <Windows.h>

static int bGameIsRunning = 1;

int main()
{
    // Game Initialization
    [Console initialization here]

    // Game Loop
    while (bGameIsRunning)
    {
        // Process Inputs
        [Input processing here]

        // Update
        [Update code here]

        // Render
        [Rendering code here]

}
    // Game Shutdown
    [Clean shutdown here]
}
```

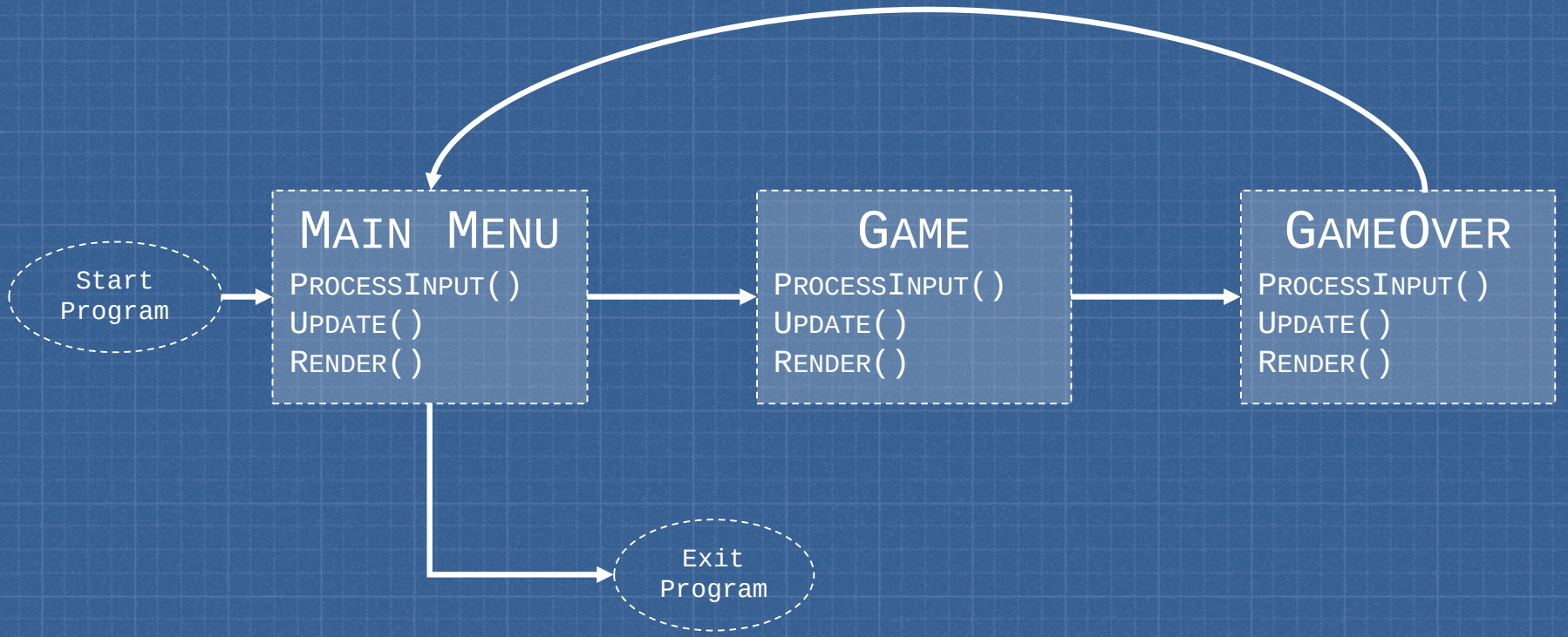
- We want
  - A Start Menu
  - The Game
  - A Game Over
- We don't want
  - Spaghetti code with nested loops
  - All the code in the Game Loop

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

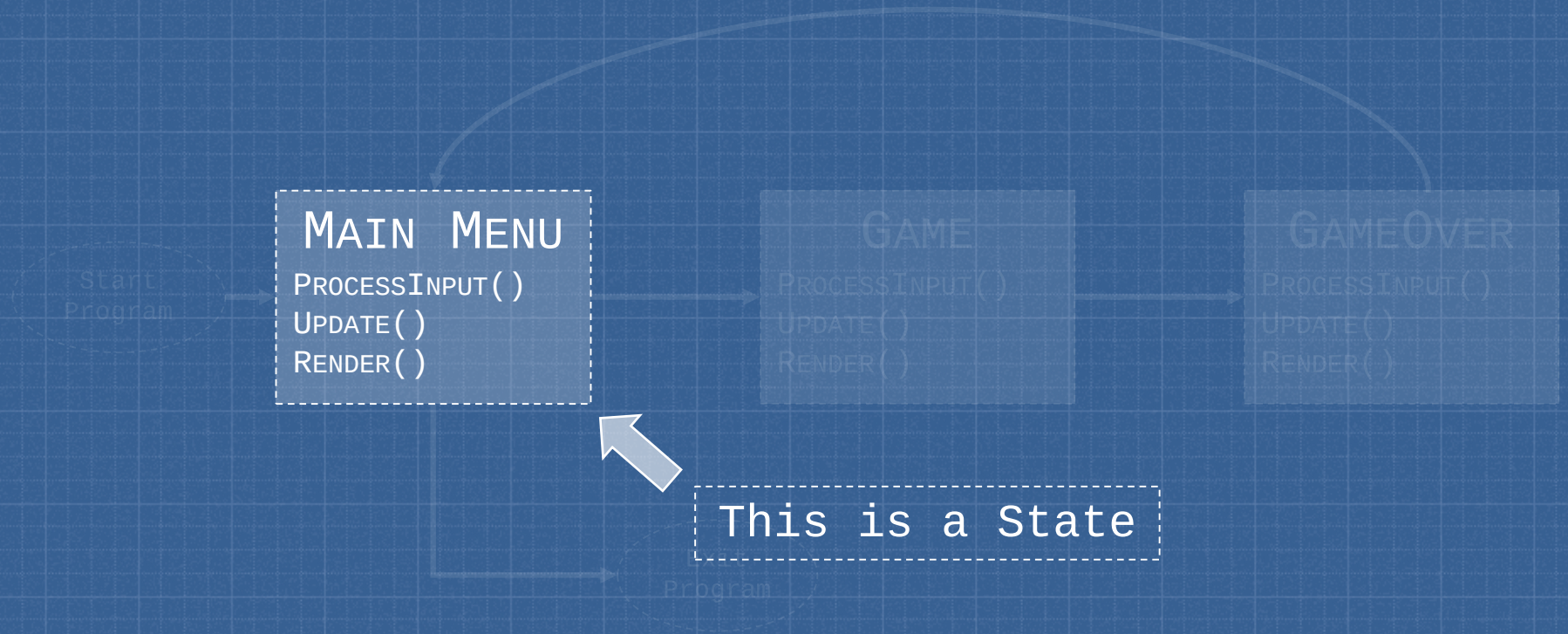
Flowchart



Flowchart


Start Program
Main Menu
ProcessInput()
Update()
Render()
Game
ProcessInput()
Update()
Render()
GameOver
ProcessInput()
Update()
Render()
Exit Program

Flowchart


Start Program
Main Menu
ProcessInput()
Update()
Render()
Game
ProcessInput()
Update()
Render()
GameOver
ProcessInput()
Update()
Render()
This is a State
Program

(Very) Simple State Machine

```
typedef enum GameState
{
    State_Default,
    State_MainMenu,
    State_Game,
    State_GameOver
} GameState;

static GameState sCurrentState = State_Default;

void ProcessInput()
{
    switch (sCurrentState)
    {
        case State_MainMenu:
            MainMenu_ProcessInput();
        break;

        case State_Game:
            Game_ProcessInput();
        break;

        case State_GameOver:
            GameOver_ProcessInput();
        break;

        default:
        break;
    }
}
```

- Define state in an Enum
  - Or constants
- Variable to keep current State
  - Change its value to change State
- Switch / case to define what to call

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

In the future

- Better ways of doing this:
  - With C function pointers
  - in C++
- Used for many things
  - AI
  - Player behaviour
  - Game Rules

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

Thanks!

Any Questions ?

Appointment:

yannick.gerber@digipen.edu